Student Data Serverless App – Project Documentation

# Project Overview

This is a fully serverless web application hosted on AWS, designed to store and retrieve student information. It leverages Amazon S3 for static website hosting, AWS Lambda for business logic, Amazon DynamoDB as the NoSQL database, and Amazon API Gateway to expose RESTful APIs.

A diagram of a software application

AI-generated content may be incorrect.

# Architecture Components

Amazon S3: Hosts a static website with HTML, CSS, and JavaScript  
Amazon API Gateway: Exposes REST API endpoints (GET and POST)  
AWS Lambda:  
 - getstudent: Retrieves all student records  
 - insertstudentData: Adds new student records  
Amazon DynamoDB: Stores student data (table: studentData)  
IAM Role: Assumed by both Lambda functions to access DynamoDB

# IAM Role

Purpose: Allows Lambda functions to read/write to DynamoDB.

Permissions:

{  
 "Effect": "Allow",  
 "Action": [  
 "dynamodb:GetItem",  
 "dynamodb:PutItem",  
 "dynamodb:Scan"  
 ],  
 "Resource": "arn:aws:dynamodb:us-east-2:<your-account-id>:table/studentData"  
}

# DynamoDB Table

Table Name: studentData

Partition Key: studentid (String)

Other Configurations: Default

# Lambda Functions

A screenshot of a computer

AI-generated content may be incorrect.

## 1. getstudent – Python 3.12

Purpose: Retrieves all records from the table

A screenshot of a computer

AI-generated content may be incorrect.

import json  
import boto3  
  
def lambda\_handler(event, context):  
 dynamodb = boto3.resource('dynamodb', region\_name='us-east-2')  
 table = dynamodb.Table('studentData')  
  
 response = table.scan()  
 data = response['Items']  
  
 while 'LastEvaluatedKey' in response:  
 response = table.scan(ExclusiveStartKey=response['LastEvaluatedKey'])  
 data.extend(response['Items'])  
  
 return data

## 2. insertstudentData – Python 3.12

Purpose: Adds new student records

import json  
import boto3  
  
dynamodb = boto3.resource('dynamodb')  
table = dynamodb.Table('studentData')  
  
def lambda\_handler(event, context):  
 student\_id = event['studentid']  
 name = event['name']  
 student\_class = event['class']  
 age = event['age']  
   
 response = table.put\_item(  
 Item={  
 'studentid': student\_id,  
 'name': name,  
 'class': student\_class,  
 'age': age  
 }  
 )  
   
 return {  
 'statusCode': 200,  
 'body': json.dumps('Student data saved successfully!')  
 }

# API Gateway

API Name: student

Type: Edge-optimized

A screenshot of a computer

AI-generated content may be incorrect.

Methods:  
 - GET: Triggers getstudent  
 - POST: Triggers insertstudentData

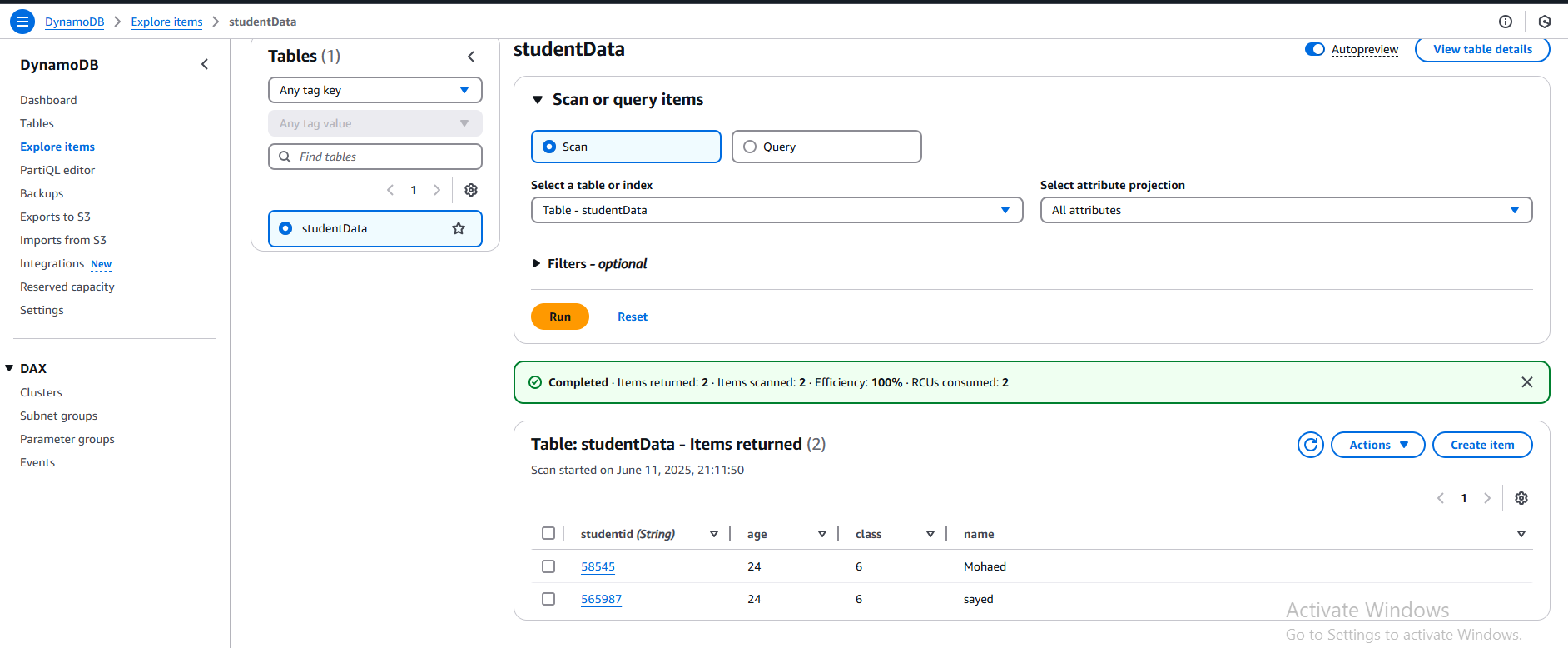
CORS: Enabled for both GET and POST methods

A screenshot of a computer

AI-generated content may be incorrect.

# S3 Static Website Hosting

Use: Hosts the frontend (HTML/JS)  
Steps:  
- Enable static website hosting  
- Paste API Gateway invoke URL in the JS code  
- Allow public access  
- Attach an inline policy to allow S3 access  
- JS code uses fetch() or axios to call the REST API



A screenshot of a student data form

AI-generated content may be incorrect.

# Future Improvements

- Enable Cross-Origin Resource Sharing (CORS) across multiple regions  
- Expand into a broader ISAAC-based distributed architecture  
- Add authentication using Amazon Cognito  
- Logging/Monitoring using CloudWatch